National College of Ireland

Configuration Manual

MSc Research Project
M.Sc. Cybersecurity

Jerry Devassy
Student ID: x19220961

School of Computing

National College of Ireland

Supervisor: Liam McCabe

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Jerry Jockey Devassy
Student ID:	X19220961
Programme:	M.Sc. Cybersecurity **Year:** 2021-2022
Module:	MSc Research Project
Lecturer:	Liam McCabe
Submission Due Date:	16/12/2021
Project Title:	Detection of Application Layer DDoS Attack by using logistic Regression
Word Count:	1241 **Page Count:** 9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jerry Jockey Devassy

Date: 16th December 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Jerry Jockey Devassy
Student ID: X19220961

1 Introduction

This study requires the usage of a dataset that includes both DDoS attack traffic and benign traffic characteristics. All irrelevant data present in the dataset such as NaaN, Negative and null values are deleted as part of the data pre-processing procedure. It is also important to do feature selection, encoding the class variables, forming a subset of datasets, developing the model, and evaluating the model. The setup manual's objective is to help users in installing this research project code on their system so that they may assess the study or make changes to match their individual requirements. The prerequisites and environment set up section povides comprehensive information for constructing a project setting as well as a list of need for reproducing the study results. The Code Execution section includes the completed created code as well as the parameters for modifying various aspects of the project.

2 Requirement

To implement the project, a system that supports all the essential tools and related settings is required, so that the implemented project runs sucessfully with all of its supported dependencies. If the system supports all of the essential tools, there will be no issues running the project resulting in a better outcome.

2.1 System Requirement

The machine learning process consumes very high resources on the host system. As a result, the hardware configuration on the system in which the project is downloaded must be capable of doing such duties. The system's minimal requirements are as follows:

For Windows:

- CPU: Intel i3 5th Gen and above
- RAM: 8GB DDR4 and above
- Storage: 20 GB free space in HDD

For iOS:

- Processor: 1.1GHz dual core Intel Core i3
- Memory: 8GB LPDDR4X onboard memory
- Storage: 128GB PCIe-based SSD

2.2 Machine Requirement

- MS Excel – for analysing the dataset
- Proper working Internet Connection
- Web Browsers - Safari / Google Chrome

2.3 Software Requirements

- Anaconda Navigator- Jupyter Notebook("Anaconda | Choose Your Anaconda IDE Adventure," 2021)

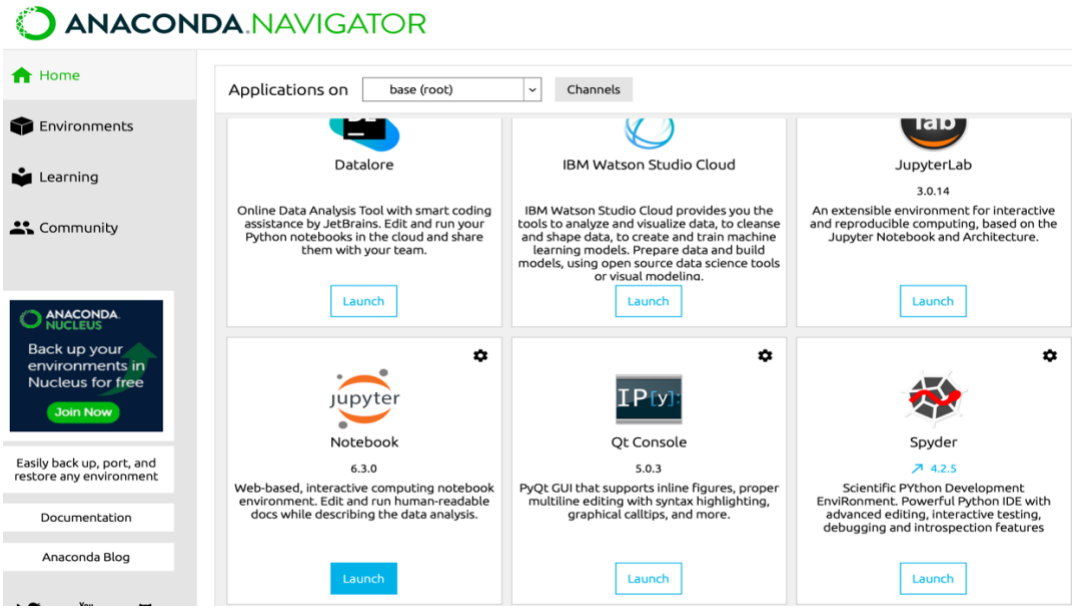


Figure 1: Anaconda Navigator- Jupyter Notebook

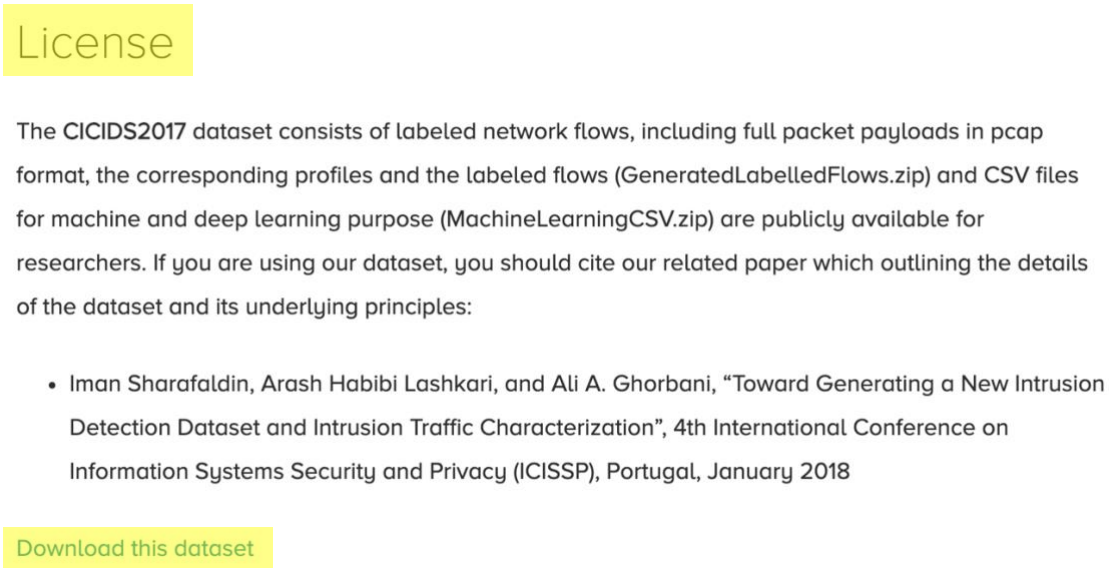
- Python 3- It is an open-sourced software available to download. We used python 3 as it has better features available as compared to python 2("Download Python," 2021).

Figure 2: Python Download

3 Dataset

The CICIDS 2017 dataset was used in this research. The dataset is a CSV file having normal traffic and four different DDoS attack traffic captured from the web traffic. This dataset is open sourced and is available to download from the CIC Website(“IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB,” 2017).

Figure 3 shows the license of the dataset and how to download the dataset. however, it will require a strong internet connection as the size of the dataset is 2GB. Figure 2 shows the sample of dataset in the excel sheet.



License

The **CICIDS2017** dataset consists of labeled network flows, including full packet payloads in pcap format, the corresponding profiles and the labeled flows (GeneratedLabelledFlows.zip) and CSV files for machine and deep learning purpose (MachineLearningCSV.zip) are publicly available for researchers. If you are using our dataset, you should cite our related paper which outlining the details of the dataset and its underlying principles:

- Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

Download this dataset

Figure 3: Dataset License and Download representation

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Destination I	Flow Duratic	Total Fwd Pa	Total Backwa	Total Length	Total Length	Fwd Packet L	Fwd Packet L	Fwd Packet L	Fwd Packet L	Bwd Packet L	Bwd Packet L	Bwd Packet L	Bwd Packet L	Flow Bytes/s	Flow Packets	Flow IAT Me	Flow IAT Std	Flow IAT Ma
2	80	38308	1	1	6	6	6	6	6	0	6	6	6	0	313.2505	52.208416	38308	0	38308
3	389	479	11	5	172	326	79	0	15.636364	31.449238	163	0	65.2	89.278777	1039666	33402.923	31.933333	25.510409	73
4	88	1095	10	6	3150	3150	1575	0	315	632.56164	1575	0	525	813.3265	5753424.7	14611.872	73	204.96097	810
5	389	15206	17	12	3452	6660	1313	0	203.05882	425.77847	3069	0	555	977.48034	665000.66	1907.1419	543.07143	2519.9314	13391
6	88	1092	9	6	3150	3152	1575	0	350	694.50972	1576	0	525.33333	813.8429	5771062.3	13736.264	78	207.00093	794
7	389	433	11	4	172	326	79	0	15.636364	31.449238	163	0	81.5	94.108094	1150115.5	34642.032	30.928571	27.744616	74
8	88	1088	9	6	3150	3152	1575	0	350	694.50972	1576	0	525.33333	813.8429	5792279.4	13786.765	77.714286	210.37711	805
9	80	579225	132	150	160	320799	160	0	1.2121212	13.926212	4344	0	2138.66	831.83483	554118	486.85744	2061.2989	12214.526	94632
10	49666	3	2	0	12	0	6	6	6	0	0	0	0	0	4000000	666666.67	3	0	3
11	49413	4	3	0	18	0	6	6	6	0	0	0	0	0	4500000	750000	2	1.4142136	3
12	88	1080	9	6	3130	3124	1565	0	347.77778	690.10013	1562	0	520.66667	806.61333	5790740.7	13888.889	77.142857	212.11458	811
13	389	5008522	39	26	5996	10284	403	0	153.74359	139.30178	1173	126	395.53846	344.60426	3250.4599	12.977881	78258.156	387630.69	2673701
14	3268	4955405	33	24	4990	5060	403	0	151.21212	120.14698	242	72	210.83333	63.434778	2028.0885	11.502592	88489.375	415887.63	2716857
15	443	274441	12	14	5763	16108	3425	0	480.25	1016.5899	3121	0	1150.5714	863.04031	79692.903	94.738031	10977.64	19256.034	52399
16	443	4	2	0	91	0	85	6	45.5	55.861436	0	0	0	0	22800000	500000	4	0	4
17	49666	24367839	11	6	920	912	224	0	83.636364	110.122	328	0	152	147.85939	75.181061	0.6976409	1522989.9	6037953.5	24200000
18	135	24368732	11	6	686	560	168	0	62.363636	80.651439	172	0	93.333333	77.755171	51.131097	0.6976153	1523045.8	6040744.1	24200000
19	49666	47	1	3	6	18	6	6	6	0	6	6	6	0	510638.3	85106.383	15.666667	22.810816	42
20	135	60	1	3	6	18	6	6	6	0	6	6	6	0	400000	66666.667	20	24.758837	48
21	49441	130	1	2	6	12	6	6	6	0	6	6	6	0	138461.54	23076.923	65	43.84062	96
22	443	8356382	62	92	182818	36159	23360	0	2948.6774	4935.9787	2920	0	393.03261	696.92525	26204.762	18.429028	54616.876	459328.75	5549208
23	443	219614	8	7	1691	4635	773	0	211.375	288.10114	1460	0	662.14286	644.25963	28805.085	68.301657	15686.714	23498.077	54428
24	443	401498	7	7	750	6161	341	0	107.14286	137.13549	1460	0	880.14286	654.42836	17213.037	34.869414	30884.462	46810.521	99732
25	443	76	3	0	97	0	85	6	32.333333	45.610671	0	0	0	0	1276315.8	39473.684	38	15.556349	49
26	44459	28460	14	9	20526	42	2920	0	1466.1429	1134.6311	6	0	4.6666667	2.6457513	722698.52	808.15179	1293.6364	4788.9565	22024
27	13792	29	1	1	6	6	6	6	6	0	6	6	6	0	413793.1	68965.517	29	0	29

Figure 4: Preview of Dataset

4 Required Packages and Imports

The model was implemented on jupyter notebook and python code was used. The Python code includes packages and imports that are listed below:

- Matplotlib 3.1.2
- Numpy 1.18.0
- scikit-learn 0.22
- sklearn
- pandas
- mpl_toolkits
- Seaborn
- train_test_split
- Sklearn.metrics
- Logisticregression

5 Execution of Code

Below are the few steps to run the code

1. Download the zip file with datasets and python files
2. Unzip the files into a folder
3. Open Jupyter notebook through anaconda navigator GUI
4. Open .ipynb file in Jupyter (DDoS_Detection_thesis)
5. Run the code, either step by step or whole code at the same time.

6 Evaluation of Code

There are multiple sections in which the whole model is implemented. Firstly, the whole data is sent to feature selection in which correlation coefficient method is used to select the best possible features and later these features is sent to logistic regression model to train and test the model and to classify if the traffic is an attack or a normal traffic.

6.1 Feature Selection and Data Processing

For feature selection, Correlation Coefficient method is used to select the best features that are dependent to each other. In Figure 5, function for plotting of correlation matrix is written but as the size of the dataset is large the plotting shows the graph but has no clear values in it. Figure 5 shows the function of correlation matrix in which it is implemented.

```
In [211]: # Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    #filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique va
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN or constant columns ({df.shape[1]}) is less
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor='w', edgecolor='k')
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for ', fontsize=15)
    plt.show()
```

Figure 5: Function for Correlation matrix

The preceding code is used to generate the correlation function, which is called by giving the data frame and the graph width in it. Figure 6 shows how the function is called and the graph width that is mentioned in it.

```
In [216]: plotCorrelationMatrix(ddos_test_df1, 19)
```

Figure 6: Plotting the Correlation function

The Output of the plotting gives the features that are strongly in relation with the target variable (variable that states whether it is legit traffic or attack traffic). Since the data was too large, the plotting matrix did not show proper data in it and therefore in Figure 7, A function has been implemented to retrieve the features which are highly correlated with each other against the target variable (Label) are called through the function itself which lists down the features in the output.

```
In [217]: # with the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feature

def correlation
correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr

In [218]: corr_features = correlation(ddos_df1, 0.99)
len(set(corr_features))

Out[218]: 21

In [219]: corr_features

Out[219]: {' Average Packet Size',
 ' Avg Bwd Segment Size',
 ' Avg Fwd Segment Size',
 ' Bwd Header Length',
 ' ECE Flag Count',
 ' Fwd Header Length',
 ' Fwd Header Length.1',
 ' Fwd IAT Max',
 ' Idle Max',
 ' Idle Min',
 ' SYN Flag Count',
 ' Subflow Bwd Bytes',
 ' Subflow Bwd Packets',
 ' Subflow Fwd Bytes',
 ' Total Backward Packets',
 ' Total Length of Bwd Packets',
 ' act_data_pkt_fwd',
 'Fwd IAT Total',
 'Fwd Packets/s',
 'Idle Mean',
 'Subflow Fwd Packets'}
```

Figure 7: List of Features

6.2 Using Logistic Regression to classify the Data

Since the dataset was huge (count of traffic in dataset was around 6lakhs), we had to create the subset of dataset from the original one which was performed in a random manner and imported the same in the code which is listed in Figure 8.

```
In [66]: #lregression.drop('Unnamed: 0', inplace=True, axis=1) # dropiing unnamed column as it is added as index

         lregression = pd.read_csv('Dataset_DDoS_10kvalues.csv')
         lregression.dataframeName = 'Dataset_DDoS_10kvalues.csv'

         nRow, nCol = lregression.shape
         print(f'There are {nRow} rows and {nCol} columns')

         There are 10000 rows and 20 columns
```

```
In [67]: # checking if the dataset has na values
         # lregression.isna()
         lregression.dropna ()

         lregression.head()
         lregression
```

Out[67]:

	Total Backward Packets	Total Length of Bwd Packets	Fwd IAT Total	Fwd IAT Max	Bwd Header Length	Fwd Packets/s	SYN Flag Count	ECE Flag Count	Average Packet Size	Avg Fwd Segment Size	Avg Bwd Segment Size	Fwd Header Length	Subflow Fwd Packets	Subflow Fwd Bytes	Subflow Bwd Packets
0	17	10393	60600000	10100000	344	0.297233	0	0	344.342857	92.166667	611.352941	372	18	1659	17
1	6	11632	12500000	12500000	200	0.457574	0	0	861.285714	53.250000	1938.666667	264	8	426	6

Figure 8: Data fed into Logistic Regression

In Figure 9, StandardScaler functions is used because in StandardScaler the mean is removed, and each feature/variable is scaled to unit variance. The data is divided into X – independent variables and y – target variables, which are subsequently utilized by the logistic regression classifier to train and test the model.

```
In [71]: # Feature Scaling - which might help to improve the performance of the algorithm
         #from sklearn.preprocessing import StandardScaler

         scalar = StandardScaler()

         X_train = scalar.fit_transform(X_train)
         X_test = scalar.transform(X_test)
```

```
In [461]: # Fitting the Logistic Regression into the Training set
          #from sklearn.linear_model import LogisticRegression

          #model_lg = LogisticRegression()
          model_lg = LogisticRegression(solver='lbfgs', max_iter=5000)
          model_lg.fit(X_train,Y_train)

          # Predicting the test set results
          Y_test_pred = model_lg.predict(X_test)
          Y_test_pred

Out[461]: array([1, 0, 0, ..., 0, 1, 0], dtype=int64)
```

```
In [463]: model_lg.score(X_test, Y_test)

Out[463]: 0.8283333333333334
```

Figure 9: Model Training and Testing

The model's output is assessed in terms of an accuracy score, which is computed using the confusion matrix. Figure 10 clearly depicts the confusion matrix and accuracy obtained while training the model using logistic regression.

```
In [464]: # Making the Confusion Matrix

          #from sklearn.metrics import accuracy_score,confusion_matrix

          confusion_matrix(Y_test, Y_test_pred)

Out[464]: array([[1341,  185],
                 [ 330, 1144]], dtype=int64)

In [465]: accuracy_score (Y_test,Y_test_pred)

Out[465]: 0.8283333333333334
```

Figure 10: Accuracy and Confusion Matrix

7 References

Anaconda | Choose Your Anaconda IDE Adventure: Jupyter, JupyterLab,… [WWW Document], 2021 . Anaconda. URL https://www.anaconda.com/blog/choose-your-anaconda-ide-adventure-jupyter-jupyterlab-or-apache-zeppelin (accessed 12.14.21).

Download Python [WWW Document], 2021 . Python.org. URL https://www.python.org/downloads/ (accessed 12.14.21).

IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB [WWW Document], 2021. URL https://www.unb.ca/cic/datasets/ids-2017.html (accessed 12.13.21).